CLEF NewsREEL 2017: Contextual Bandit News Recommendation

Yu Liang[1], Babak Loni[1] and Martha Larson[1,2]

[1] Delft University of Technology, Netherlands
[2] Radboud University Nijmegen, Netherlands
Y.Liang-2@student.tudelft.nl, b.loni@tudelft.nl, m.a.larson@tudelft.nl

Abstract. In the CLEF NewsREEL 2017 challenge, we build a delegation model based on the contextual bandit algorithm. Our goal is to investigate whether a bandit approach combined with context extracted from the user side, from the item side and from user-item interaction can help choose the appropriate recommender from a recommender algorithm pool for the incoming recommendation requests. We took part in both tasks: NewsREEL Live and NewsREEL Replay. In the experiment, we test several bandit approaches with two types of context features. The result from NewsREEL Replay suggests that delegation model based on the contextual bandit algorithm can improve the click through rate (CTR). In NewsREEL Live, a similar delegation model is implemented. However, the delegation model from NewsREEL Live is trained by the data stream from NewsREEL Replay. This is due to the fact that the low volume of data received from the online scenario is not enough to support the training of the delegation model. For our future work, we will add more recommender algorithms to the recommender algorithm pool and explores other context features.

Keywords: Recommender System, Context, Contextual Bandit, News, Evaluation

1 Introduction

The CLEF NewsREEL Challenge [1, 2] is a challenge in which participants are asked to provide effective real-time news recommendation. The challenge provides participants with a platform to evaluate their recommender algorithms with millions of real-world users. We took part in both tasks [3]: Task 1 (NewsREEL Live) [4] and Task 2 (NewsREEL Replay) [5]. In the Live task, the recommendation requests are distributed to all participants of the challenge and thus not all traffic is sent to a single participant. This makes it more challenging than the Replay task. In the Live task, we use a trained delegation model that is implemented for the Replay task.

Our work was initially inspired by Lommatzsch and Albayrak, who, in their work [6], combine several recommender algorithms with one delegation model responsible for delegating the incoming recommendation request to the appropriate recommender algorithm. In our work, we also build a similar delegation

model with the objective to improve recommendation performance, in other words, to maximize Click Through Rate (CTR). We find that a multi-armed bandit (MAB) algorithm is suited for the purpose. Multi-armed bandit algorithms address the problem of a gambler at a row of slot machines who has to decide which machine to play, in what sequence to play and how many times to play in order to maximize the rewards collecting from the slot machines. The model has been widely used in website optimization [7] for choosing the right ads for users [8] and selecting the right advertisement format [9]. All of these models involve optimization problems, e.g., selecting the suitable ad format in order to maximize the CTR received from users. Another good property of MAB algorithms is that they can continuously learn from the past, such as from user feedback, and then adjust its decisions about which arm to choose. With the feedback, it is able to balance the exploration and exploitation dilemma. Exploitation means the MAB algorithm chooses the arm with the currently best performance from the previous plays, while exploration means it chooses the arm with lower performance in order to learn more from these arms. This kind of trade-off is missing in A/B testing, in which a pure exploration is applied at the initial phase followed by a pure exploitation in a very long period [7]. The exploration and exploitation trade-off makes MAB algorithm works well for the news recommendation [10], in which it is able to recommend popular items most of the time and also explore the news items in the long tail. Further, combined with context, the model works even better. Context is proved to be useful for the optimization [7–10].

However, our consideration is not directly related to news items, but the recommender algorithms. Given the effective recommender algorithms from previous years, we believe the contextual bandit algorithm is able to combine them in a wise way and ultimately improve the recommendation performance. Our research interest comes with the assumption that given the context of a user, item and user-item interaction, contextual bandit algorithms can help choose appropriate recommender algorithm for each recommendation request and thereby improve the CTR.

This paper is structured as follows: in Section 2, we discuss the work related to the contextual multi-armed bandit algorithm. In Section 3, we introduce the method that we implemented in this work. Section 4 describes our evaluation methods, including NewsREEL Replay and NewsREEL Live. In Section 5, we analyze the evaluation results. In the last section, we summarize the results and discuss future work.

2 Background and Related Work

2.1 Multi-armed Bandit Algorithms

epsilon-Greedy Epsilon-Greedy is the simplest algorithm to solve the multi-armed bandit problem. The only parameter in the model is ϵ. In the standard settings, ϵ is fixed. With probability $1 - \epsilon$, the policy plays the currently best

arm, and with probability ϵ, it plays a random arm. The algorithms also have several variations, e.g., ϵ-decreasing in which ϵ decreases over time [11].

Upper Confidence Bound Unlike epsilon-greedy, UCB is not only concerned about the reward from the previous plays, but also about how much it knows about the available arms. At the initial phase, UCB explores each arm at least once. After the initial phase, each time it plays the arm with the highest value from the formula $r_\alpha + b$, where r_α is the estimated reward for arm α, and b is a special bonus for the arm α. One widely used UCB variation is UCB1 [12], in which b is set to be $\sqrt{\frac{2ln\sum_{i=1}^{n} t_i}{t_\alpha}}$, t_i is the number of plays arm i has been chosen, and t_α is the number of plays arm α has been chosen. In UCB, arms that are explored less in the previous plays would receive higher bonus value, while arms that are explored more would receive lower bonus value. This can prevent the under-exploration of potential rewarding arms.

Thompson Sampling Another widely used multi-armed bandit algorithm is Thompson Sampling. Thompson sampling models the probability distribution of each arm's reward. The reward of arm α follows a probability distribution with mean θ^*_α [13]. In the case where bandits are binary, a Beta-Bernoulli distribution [14] is used. The algorithms repeatedly draw a sample from the distribution of each arm, and it plays the arm with the largest value.

2.2 Contextual Bandit Algorithms

Contextual bandit algorithms combine MABs with context features. In our settings, users and news articles are always represented by feature vectors, such as domains. The contextual bandit algorithm chooses arms in such a way that the most appropriate arm for each context is played to maximize the reward. A trivial implementation is to build a MAB model for each context. Another idea is assuming there is a relationship between the expected reward and the context. We implement both ideas.

Linear UCB Linear UCB is a contextual bandit algorithm first proposed in [10]. The algorithm assumes that there is a linear relationship between the expected reward and the context. In this algorithm, the context is used to update the rewards and improve its arm selection strategy. In each trial (play) t, the expected payoff of arm α is equal to $E[r_{t,\alpha}|x_{t,\alpha}] = x_{t,\alpha}\theta^*_\alpha$, where $r_{t,\alpha}$ specifies the reward of arm α at trial t, $x_{t,\alpha}$ specifies the feature vector for arm α at trial t and θ^*_α is a coefficient that can be learned from the previous trials. In each trial, the algorithm plays the arm with the highest final score. The final score of each arm is calculated as:

$$\alpha_t = \text{argmax}_{\alpha \in \mathtt{A}} \left(x_{t,\alpha}\hat{\theta}_\alpha + \beta\sqrt{x^T_{t,\alpha}A^{-1}_\alpha x_{t,\alpha}} \right) \quad (1)$$

where $\mathtt{A}$ is the set of arms, $\hat{\theta}_\alpha$ is an estimate of coefficients, β is a hyper-parameter and A is the covariance matrix of coefficients.

3 Framework

Our framework is designed to validate the hypothesis that contextual bandit algorithms can choose the appropriate recommender algorithm from the recommender algorithm pool for the incoming recommendation requests given context from users and items. In NewsREEL Replay, we consider two types of training: the training of the delegation model and the training of the recommender algorithms. The delegation model is trained and updated in batches (a more detailed discussion is presented in Section 3.2). In contrast, the recommender algorithms are trained and are able to update continuously regardless of the batches. The delegation model implemented in the Live task is also trained in the same way, but the data stream for training is from the offline dataset due to the low volume received from the online scenario. The recommender algorithms in the Live scenario also update continuously.

3.1 Recommender Algorithms

We build up our delegation recommender based on the recommender developed by recommenders.net. The recommender algorithm pool consists of five simple recommender algorithms:

- *Recent*: Recommend the most recently created or updated articles. For the realization, a ring buffer is implemented to store the most recently created or updated articles. The size of the ring buffer is set to 100. When there is a new item, it will be inserted into the ring buffer. If the ring buffer reaches its size, it will drop the least recent news articles.
- *Path*: Given the news articles that the user is currently reading, recommend the most popular[3] news articles requested next by users, referring to the *Most Popular Sequence* in the work [6]. In the default settings, for each news article, 100 most recently requested articles are kept. The popularity scores are then computed from the 100 articles.
- *Click*: Similar to the algorithm *Path*, but the popularity algorithm considers only the clicks[4] of the user.
- *Imp*: Similar to the algorithm *Path*, but the popularity algorithm considers only the impressions from users.
- *PRCate*: Popularity and category based Recommender. Recommend the news article with most impressions and clicks within a certain category. The popularity scores only consider 100 most recently requested news articles within the category.

[3] The popularity is computed from the weighted combination of impressions and clicks.

[4] A click is different from an impression: a click means the user clicks on the recommendation, while an impression means the user clicks on articles that were not necessarily recommended [15]

3.2 Delegation Framework

Our proposed delegation framework is trained and updates only in the Replay task. The low volume of data from the Live task is not enough for training such a framework. However, in NewsREEL Live, we implement the delegation model trained from the NewsREEL Replay. More details about NewsREEL Live can be found in Section 4.1. It is not trivial to implement MAB algorithm in the real-world settings. We first define the reward. In our settings, the reward is defined as clicks received from recommendations. If a user clicks on a recommendation, the reward is 1, otherwise, the reward is 0. Therefore, the reward is a binary value. Given the context, the recommender algorithm, and its corresponding reward, the delegation model is able to update. However, unlike what is implemented in [9, 10], where rewards can be obtained immediately after each action, rewards in real-world settings are delayed. Specifically, in our case, there would be hundreds of incoming recommendation requests between a recommendation and its corresponding reward. It is unrealistic to suspend the delegation model before receiving the reward or to update the whole delegation model each time a reward is observed. We implement a similar batch update as used in the work [8] for our delegation model. The basic underlying idea is to update the delegation model in batches, in other words, the model updates only after receiving enough amount of reward.

Batch update and assignment The whole recommendation process is divided into batches. The contextual bandit algorithm is updated at the end of each batch with the rewards collected in each batch, as illustrated in Figure 1. After each update, the delegation model will preassign appropriate recommender for each context to serve incoming requests in the next batch.

Initialization In the first batch, there is no historical data from which the contextual bandit algorithm can learn the model. Therefore, at the initial phase of recommendation, recommender algorithms are randomly selected to serve the incoming recommendation requests. At the end of the first batch, the contextual bandit algorithm is able to update for the first time with the rewards collected in the first batch.

Rewards Rewards are defined as 'clicks' received from the recommendations. However, in both tasks, we cannot directly track whether our previous recommendations are successful or not. Making use of the evaluator from NewsREEL Replay, the reward is defined, for a given recommendation, based on whether the user interacts with the item in the next 10 minutes.

3.3 Exploiting Context

We exploit context with two different methods. In the first method, the context is used to filter the model. For example, if the recommendation request belongs

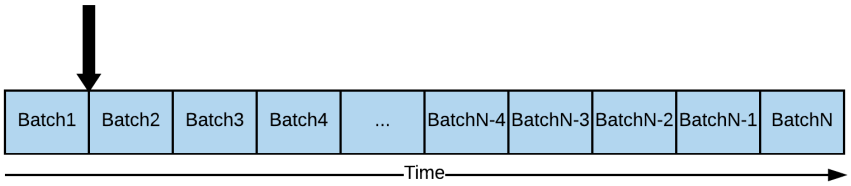

Fig. 1. Illustration of updates and the end of each batch.

to a domain A, the method chooses the model that is trained from items in A. We refer to this method as *context pre-filtering*. Our second approach exploits context for training the model. We refer to this method as *context modeling*. We use the Linear UCB algorithm [10] to exploit context in the model. We use a special case of general contextual bandit algorithm known as *K-armed bandit*, where the set of arms remains the same in all trials. In our framework, the five algorithms introduced earlier in this section are served as the arms. The score of arms is calculated according to Eq.(1) where the feature vectors $x_{t,\alpha}$ are built based on the given context.

For the contextual bandit algorithm, it is important to find useful context features. Unlike other work [8, 10], where the user profile is available, we do not have any information about the user profile. Users can only be tracked by their cookies, and some users even do not allow any tracking. All contextual information is encoded in the incoming messages. In addition, not all context features can contribute to the performance of recommender algorithms. We consider two context features, one is *Domain*, which is also considered as an important context feature in the work [6] and the other is *User-Domain Impressions*. *Domain* refers to different publishers, and *User-Domain Impressions* refers to the number of impressions the user has with the domain.

Figure 2 shows the recommender algorithms' performance with respect to different context features using the CTR metric. In domain '418', a general news portal, the recommender *Click* performs the best. This recommender also shows a good performance in domain '1677', which is also a general news portal. However, the popularity based algorithm does not perform well in domain '35774', a sports news portal. The recommender algorithm *Recent* shows the best performance in this domain. This can be explained by the fact that people always prefer the newest sports news. Also, the bad performance of *PRCate* might be due to the reason the sports domain is already very specific, and further specific category would not help. The findings are consistent with the findings in previous work. Figure 3 shows the algorithms performance with respect to the number of user-domain impressions in domain '418'. Normally, with the increase of the number of user-domain impressions, CTR should also be increased as users with more interactions with the publishers would be more willing to click

on the recommendations. However, there are some exceptions. With the increase of *User-Domain Impressions*, CTR observed from the *Click* algorithm experience a sharp decrease. The decrease can be explained by the fact people with enough interactions with the publisher have already seen the popular articles and prefer the newest news. In *context modeling*, the categorical context features are encoded by one-hot coding. The context *Domain* is already a categorical feature by itself. For the context *User-Domain Impressions*, we implement some transformation. Specifically, we classifies the number of *User-Domain Impressions* into three groups: $0 \leq$ number of *User-Domain Impressions* < 5, $5 \leq$ number of *User-Domain Impressions* < 10 and $10 \leq$ number of *User-Domain Impressions* < 15.

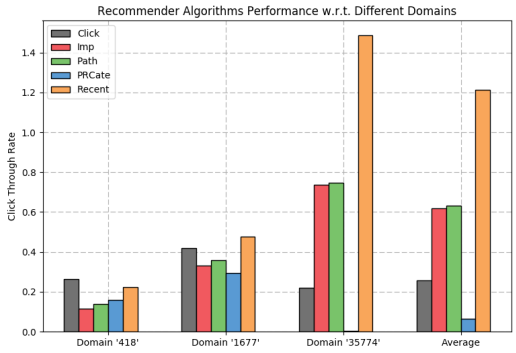

Fig. 2. Recommender Algorithms Performance w.r.t. Different Domains

4 Experiment

In both online and offline scenario, there are four types of messages: *recommendation_request*, *item_update*, *event_notification* and *error_notification*. Specifically, in the Live task, the recommender needs to give recommendations within 100ms, otherwise the recommendations will be regarded as invalid. As the data volume redirected to our recommender is too low in the Live task, the delegation model implemented online is the one with the best performance in NewsREEL Replay. Thus, in the following sections, we will first discuss our experiment in NewsREEL Replay, and then NewsREEL Live.

4.1 NewsREEL Replay

Our recommendation framework that is used in NewsREEL Replay is described in Section 3. For this task, we are provided with a month-long dataset collected

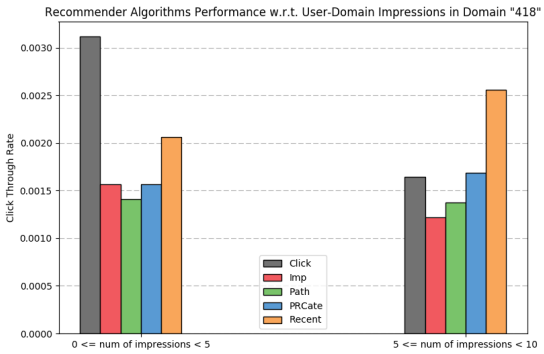

Fig. 3. Recommender Algorithms Performance w.r.t. User-Domain Impressions in Domain "418"

during February 2016. Normally, a replay of a daily dump takes about 3-4 hours. Due to the time constraints, data collected from 2016-02-01 to 2016-02-03 is replayed for evaluation. The dataset is not evenly distributed: about 75% traffic is from the domain '35774', a sports domain, followed by two general news domains '1677' and '418'. In the evaluation phase, we only consider the results from the above three domains since a majority of traffic comes from these three domains. The evaluation phase is as follows: Firstly, the daily dump from 2016-02-01 is replayed to prevent the cold-start problem. Then the evaluation phase is implemented in the following way: The rest of the dataset (data from 2016-02-02 to 2016-02-03) is split into N smaller mutually exclusive datasets with the same size. The dataset is replayed in batches. See Figure 1 for clarification. There are two different batch sizes, one containing 100000 line messages, and the other containing 300000 line messages. Next, as explained in Section 3.1, for each batch, rewards are computed at the end of the batch. After that, the contextual bandit algorithm is able to update the model. Finally, the chosen recommender algorithm is assigned for the incoming requests in the next batch. In Table 1,

Table 1. NewsREEL Replay Evaluation Results: Single Recommender Algorithm Baselines

Algorithm	Average CTR	Domain '418'	Domain '1677'	Domain '35774'
Random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
Path	0.632 (−2.02%)	0.137 (−26.74%)	0.357 (−10.97%)	0.748 (−0.13%)
Recent	**1.213 (88.06%)**	0.224 (19.79%)	**0.478 (19.20%)**	**1.487 (98.53%)**
Click	0.256 (−60.31%)	**0.264 (41.18%)**	0.419 (4.49%)	0.220 (−70.63%)
Imp	0.618 (−4.19%)	0.114 (−39.04%)	0.330 (−17.71%)	0.738 (−1.74%)
PRCate	0.064 (−90.08%)	0.159 (−14.97%)	0.294 (−26.68%)	0.03 (−99.60%)

results with the best performance in each domain are highlighted. In Table 2 and 3, results with similar or better performance than the results in Table 1 are also highlighted. For all algorithms, we measure their performance based on average CTR and CTR in the three domains. We also further compare the CTR with the random baseline. The percentage in the tables shows the CTR change comparing to the baseline.

Table 2. NewsREEL Replay Evaluation Results: Context Pre-filtering

Algorithm	Average CTR	CTR '418'	CTR '1677'	CTR '35774'
random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
epsilon_0.5_b3	0.954 (47.91%)	0.239 (27.81%)	0.437 (8.98%)	1.148 (53.27%)
epsilon_0.3_b3	1.070 (65.89%)	0.252 (34.75%)	0.468 (16.71%)	1.294 (72.76%)
epsilon_0.1_b3	1.163 (80.31%)	0.235 (25.67%)	0.456 (13.72%)	1.423 (89.99%)
ucb1_b3	1.197 (85.58%)	0.200 (6.95%)	0.420 (4.74%)	1.474 (96.80%)
ts_b3	**1.213** (**88.06%**)	0.224 (19.79%)	**0.480** (**19.70%**)	**1.486** (**98.40%**)
epsilon_0.5_b1	0.958 (48.53%)	0.232 (24.06%)	0.449 (11.97%)	1.151 (53.67%)
epsilon_0.3_b1	1.060 (64.34%)	0.249 (33.16%)	0.458 (14.21%)	1.283 (71.30%)
epsilon_0.1_b1	1.160 (79.84%)	**0.266** (**42.25%**)	0.456 (13.72%)	1.415 (88.92%)
ucb1_b1	1.195 (85.27%)	0.199 (6.42%)	0.405 (1.00%)	1.475 (96.93%)

[1] Algorithm labels: *epsilon_0.5_b3:* epsilon_greedy with $\epsilon = 0.5$ with context *Domain* and batch size 300000, *ucb1_b3:* UCB1 with context *Domain* and batch size 300000, *ts_b3:* Thompson Sampling with context *Domain* and batch size 300000

With the context given in Section 3.2, we propose two different delegation models in the Replay task. In the first method, referred as *context pre-filtering*, we only consider the context *Domain* and implement a separate bandit model in each of the three news domains. In the second approach, referred as *context-modeling*, we encode context (*Domain* and *User-domain Impressions*) as feature vectors. Then we build a contextual bandit model based on the linucb (Linear UCB) model. The results of the simple baseline recommender algorithms are presented in Table 1, the results of the context pre-filtering model in Table 2, and the results of the context-modeling approach is listed in Table 3. The random algorithm, which randomly selects a recommender algorithm to serve the incoming requests, is set as the baseline, referred to as *random* in Table 1. The names of the contextual bandit algorithms in Table 2 contain *algorithm_hyperparamter value_batch size*. There are three different bandit algorithms in context pre-filtering: epsilon-greedy (epsilon), ucb1 and Thompson Sampling (ts). For epsilon-greedy, we tried three different ϵ value: 0.5, 0.3 and 0.1. In the experiment, we also tried two batch sizes: b3 means each batch containing 300000 messages and b1 indicating 100000 messages. Contextual bandit algorithms in Table 3 are formatted as *algorithm_hyperparamter value_context used_batch size*. The only algorithm used in context-modeling is linear UCB. In the model, linear UCB is implemented with three different β value in Eq.(1): 1,

Table 3. NewsREEL Replay Evaluation Results: Context-Modeling

Algorithm	Average CTR	CTR '418'	CTR '1677'	CTR '35774'
linucb_1_d_b3	1.202 (86.36%)	0.220 (17.65%)	0.403 (0.50%)	**1.483 (98.00%)**
linucb_0.5_d_b3	1.207 (87.13%)	0.246 (31.55%)	0.423 (5.49%)	**1.483 (98.00%)**
linucb_0.2_d_b3	1.204 (86.67%)	0.257 (37.43%)	0.414 (3.24%)	**1.484 (98.13%)**
linucb_1_di_b3	1.211 (87.75%)	0.259 (38.50%)	0.451 (12.47%)	**1.485 (98.26%)**
linucb_0.5_di_b3	1.206 (86.98%)	**0.279 (49.20%)**	0.415 (3.49%)	**1.485 (98.26%)**
linucb_0.2_di_b3	**1.214 (88.22%)**	**0.268 (43.32%)**	0.467 (16.46%)	**1.485 (98.26%)**
linucb_1_d_b1	0.992 (53.80%)	0.244 (30.48%)	0.452 (12.72%)	1.180 (57.54%)
linucb_0.5_d_b1	1.205 (86.82%)	0.231 (23.53%)	0.457 (13.97%)	1.478 (97.33%)
linucb_0.2_d_b1	1.196 (85.43%)	**0.263 (40.64%)**	0.393 (−2.00%)	1.478 (97.33%)
linucb_1_di_b1	1.196 (85.43%)	0.253 (25.29%)	0.395 (−1.50%)	1.478 (97.33%)
linucb_0.5_di_b1	1.196 (85.43%)	0.207 (10.70%)	0.426 (6.23%)	1.477 (97.20%)
linucb_0.2_di_b1	1.197 (85.58%)	0.150 (−19.79%)	0.456 (13.72%)	1.479 (97.46%)

[1] Algorithm annotation: *linucb_1_d_b3:* LinUCB with $\beta = 1$ with context *Domain* and batch size 300000, *linucb_1_di_b3:* LinUCB with $\beta = 1$ with context *Domain*, *User-Domain Impressions* and batch size 300000

0.5 and 0.2, and two context: *Domain* and *User-Domain Impressions*. In Table 2 and Table 3, *d* stands for the context *Domain*. In Table 3, *di* indicates the use of both contexts.

4.2 NewsREEL Live

The online evaluation of NewsREEL Live is from 24 April to 7 May 2017. The Open Recommendation Platform (ORP) is responsible for redirecting the traffic from the publishers and monitoring different recommender algorithms' performance. With ORP, participants are able to deploy their recommender services and receive recommendation requests. The low volume of data refers to the small number of recommendation requests, user clicks and user impressions, redirected to our recommender from plista. Comparing with the data stream in the Replay dataset, the data stream redirected in the Live task consists only a small portion of the whole plista data stream and is not enough to support training or updating of our contextual bandit algorithm. In the Live task, we deployed the best delegation recommender model trained from the NewsREEL Replay, referring to *linucb_0.2_di_b3* in Table 3, to serve the incoming recommendation requests. In addition, a portion of data from the Live task comes from a domain that is not included in the Replay dataset. This also gives rise to a challenge for us. Our solution is that for all the recommendation requests from that domain, we redirect them to the *PRCate* recommender. Table 4 shows our online evaluation results. There are two types of impressions created by ORP. The first type considers the impressions from the recommendation list, which are referred to as 'widget impressions', and the second type considers the impressions from individual items, which are referred to as 'item impressions'. ORP also creates a

Table 4. NewsREEL Live Evaluation Results from 24 April to 7 May 2017 (delegation model: *linucb_0.2_di_b3* trained from offline)

Metric	Click Num	Impression Num	CTR
Widget	747	60814	0.012
Individual item	747	192207	0.0039

'click' for each click on the recommended articles from users. The final metric for the recommender performance is measured by the CTR, which equals to the number of clicks divided by the number of impressions.

5 Discussion

5.1 NewsREEL Replay

The NewsREEL Replay evaluation results are summarized in Table 1, Table 2 and Table 3. Table 1 shows the recommendation performance of the five basic recommenders from which the delegation model is able to choose an appropriate recommender. A random algorithm, which randomly selects a recommender for the incoming recommendation requests, is served as the baseline. Table 2 shows the evaluation results from the contextual bandit delegation model in which a MAB model is set up for each context. Table 3 shows the evaluation results from the contextual bandit delegation model in which the linear UCB model is implemented. The only context in Table 2 is *Domain*, while in Table 3, two contexts *Domain* and *User-Domain Impressions* are considered. From both Table 2 and Table 3, we can see the evaluation results from the delegation model outperform the random baseline. This indicates the delegation model is able to learn from the past and chooses appropriate recommender for the incoming recommendation requests in order to maximize the CTR.

To further explain the results, we first describe two facts with respect to the domain '35774': Firstly, as explained in the previous section, about 70% of the traffic is from this domain. Therefore, a CTR increase in domain '35774' leads to an increase of average CTR. Secondly, as shown in Table 1, the recommender *Recent* performs much better than any other recommender in this domain. For the performance of different MAB algorithms, UCB1 works better than epsilon-greedy in domain '35774', but worse in domain '418' and domain '1677'. This result is a little bit unexpected since UCB1 explores under the guidance of a confidence bound [10], while epsilon-greedy explores randomly without any guidance. The reason might be the parameter that is used to balance the exploration and exploitation is fixed in UCB1. In the future, we will try UCB with other value of the parameter. Thompson Sampling also works well in the evaluation, and in domain '1677' it shows the best performance. For domain '35774', further exploration does not benefit the CTR since the *Recent* recommender always performs best in the domain. The exploration leads to a CTR decrease in domain '35774' but CTR increases in domain '418' and domain '1677'. For

epsilon-greedy, smaller ϵ value shows better results. From the side of epsilon-greedy, exploring too much is also not a good idea. The fixed parameter ϵ is served to balance the exploration and exploitation. In Table 3, in addition to the context *Domain*, another context *User-domain Impressions*, which measures the number of interactions the user has with the domain, is also taken into consideration. The results in Table 3 are in general better than the results from Table 2. The contextual bandit delegation model based on linear UCB works better than the one in which there is a MAB model for each context. In addition, the context *User-domain Impression* is a useful context concerning CTR. Average CTR from the delegation model based on the *linucb_0.2_di_b3* algorithm even outperforms the *Recent* Recommender in Table 1. One interesting observation is that CTR does not increase in domain '35774' and '1677' indicating that the exploration is less useful, but it does increase in domain '418', in which exploration is much more useful. This is due to some recommender algorithms always achieving the best performance in some domains (e.g., *Recent* recommender in domain '35774'), making exploration is unnecessary. The findings also indicate the performance of a single recommender algorithm appears to depend on the publishers.

5.2 NewsREEL Live

For NewsREEL Live we test our recommender for 14 days. There are two challenges in the Live task: Firstly, as discussed before, the low volume of data from the ORP to our account makes it hard for us to train the contextual bandit algorithm online. A delegation model trained from the Replay task is deployed to serve the incoming recommendation requests. Secondly, publishers in the Live task are different from the publishers in the Replay task. In NewsREEL Live, a portion of requests comes from domain '17614', which is not included in the Replay dataset. As the result, the corresponding requests from that domain are delegated to the *PRCate* recommender. The two challenges might lower our recommendation performance in the Live task. In total 19 teams which participate in NewsREEL Live, our delegation model ranked 10th.

5.3 Conclusion and Future Work

In conclusion, our results have verified our assumption the contextual bandit algorithm is able to select the appropriate recommender algorithm given the context. However, the balance between exploration and exploitation is important. If one particular recommender algorithm is performing significantly better, then exploitation does not necessarily help. For future work, firstly, we will add more recommender algorithms to our recommender algorithm pool, such as collaborative filtering since the performance of the contextual bandit algorithm is to some extent limited by the single recommender algorithm. Secondly, we will explore more contexts in order to further improve the recommendation CTR. Last but not least, we would like to evaluate the contextual bandit algorithm on a dataset with a more equally distributed traffic among different domains.

References

1. Hopfgartner, F., Brodt, T., Seiler, J., Kille, B., Lommatzsch, A., Larson, M., Turrin, R., Serény, A.: Benchmarking news recommendations: The CLEF NewsREEL use case. In: ACM SIGIR Forum, ACM (2016) 129–136
2. Lommatzsch, A., Kille, B., Hopfgartner, F., Larson, M., Brodt, T., Seiler, J., Özgöbek, Ö.: CLEF 2017 NewsREEL overview: A stream-based recommender task for evaluation and education. In: Proceedings of CLEF 2017, Dublin, Ireland, 2017. (2017)
3. Kille, B., Lommatzsch, A., Gebremeskel, G.G., Hopfgartner, F., Larson, M., Seiler, J., Malagoli, D., Serény, A., Brodt, T., De Vries, A.P.: Overview of NewsREEL'16: Multi-dimensional evaluation of real-time stream-recommendation algorithms. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2016) 311–331
4. Hopfgartner, F., Kille, B., Lommatzsch, A., Plumbaum, T., Brodt, T., Heintz, T.: Benchmarking news recommendations in a living lab. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2014) 250–267
5. Kille, B., Lommatzsch, A., Turrin, R., Serény, A., Larson, M., Brodt, T., Seiler, J., Hopfgartner, F.: Stream-based recommendations: Online and offline evaluation as a service. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2015) 497–517
6. Lommatzsch, A., Albayrak, S.: Real-time recommendations for user-item streams. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM (2015) 1039–1046
7. White, J.: Bandit algorithms for website optimization. O'Reilly (2012)
8. Tang, L., Rosales, R., Singh, A., Agarwal, D.: Automatic ad format selection via contextual bandits. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, ACM (2013) 1587–1594
9. Li, W., Wang, X., Zhang, R., Cui, Y., Mao, J., Jin, R.: Exploitation and exploration in a performance based contextual advertising system. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2010) 27–36
10. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World Wide Web, ACM (2010) 661–670
11. Burtini, G., Loeppky, J., Lawrence, R.: A survey of online experiment design with the stochastic multi-armed bandit. arXiv preprint arXiv:1510.00757 (2015)
12. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2-3) (2002) 235–256
13. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: Advances in neural information processing systems. (2011) 2249–2257
14. Agrawal, S., Goyal, N.: Analysis of Thompson sampling for the multi-armed bandit problem. In: COLT. (2012) 39–1
15. Yuan, J., Lommatzsch, A., Kille, B.: Clicks pattern analysis for online news recommendation systems. In Balog, K., Cappellato, L., Ferro, N., Macdonald, C., eds.: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. Volume 1609 of CEUR Workshop Proceedings., CEUR-WS.org (2016) 679–690